Project Report

Arena Ticket Booking System

Leander Almonte & Luke Nwantoly

Vanier College

Programming Patterns

Sakkavarthi Ramanathan

Tuesday, December 10, 2024

Project Description

Programming Patterns Final Project: Arena Ticket Booking System

The Arena Ticket Booking System is an application where users can search and buy tickets for different events at an arena. The price of a ticket will depend on the section of the seat and the event type. They can select events like sports, concerts and spectacles. Technicians will handle the processing of a ticket by a user and handle the transactions. The user may refund a ticket and view the tickets they bought.

Classes: User Technician Arena Section Seat Event Ticket

Output: The program begins with a login screen where, depending on the credentials entered, the program will determine whether the user is a regular user or a technician. If it is a user, the program will display options for the user to pick their desired ticket(s). Once processed by the technician, the user will now happily own the ticket. When a user wants to do a refund, the program will ask which ticket they want to refund, and then the technician will execute the refund. At any time, the user may view their purchased tickets. The technician may also at any time view the total tickets sold and the users associated with those tickets. The technician can also create an Event, which will then generate a set number of tickets and add them to the list of available tickets for the users to book.

Project Features

MVC architecture

Model:

Arena, Section, Seat,

User, Technician,

Ticket, TicketSystem,

Event, BasketballGame, HockeyGame, Concert, Spectacle,

GuiModel (Database)

View:

GUIApplication

Controller:

GuiController

Singleton Pattern

To keep track of the users, technicians, tickets and Events, we used Singleton design pattern to create a single instance of the TicketSystem class that way the data is tracked consistently throughout the whole project. For example, tracking the number of available tickets, throughout the different classes that execute changes to the available tickets.

```java
 8    public  class TicketSystem {  ± LeanderAlmonte +1 *
 9       private static TicketSystem instance;  3 usages
10
11        private List<Ticket> unassignedTicket;  4 usages
12        private List<Ticket> pendingTicket ;  4 usages
13        private List<Ticket> processedTickets;  4 usages
14
15        private List<Event> events;  3 usages
16
17        private List<User> users;  5 usages
18        private List<Technician> technicians;  3 usages
19
20        private TicketSystem() {  1 usage  ± LeanderAlmonte +1 *
21            unassignedTicket = new ArrayList<>();
22            processedTickets = new ArrayList<>();
23            pendingTicket = new ArrayList<>();
24            events = new ArrayList<>();
25
26            users = new ArrayList<>();
27            technicians = new ArrayList<>();
28
29            GuiModel.loadUsers( system: this);
30            GuiModel.loadTechnicians( system: this);
31            GuiModel.loadUnsassignedTickets( system: this);
32            GuiModel.loadProcessingTickets( system: this);
33            GuiModel.loadAssignedTickets( system: this);
34            GuiModel.loadEvents( system: this);
35        }
```

Log In

Depending on the credentials, the application will check the TicketSystem's list of users and technicians to see whether the user trying to log in is a user or technician, if the credentials are invalid or nothing is entered, the system will show an output Label indicating the issue.

Arena Ticket Booking System

Arena Ticket Booking System

French

Username

Log In

Arena Ticket Booking System

French

Username

Invalid Username and Password

Log In

Arena Ticket Booking System

French

invalid

•••••••

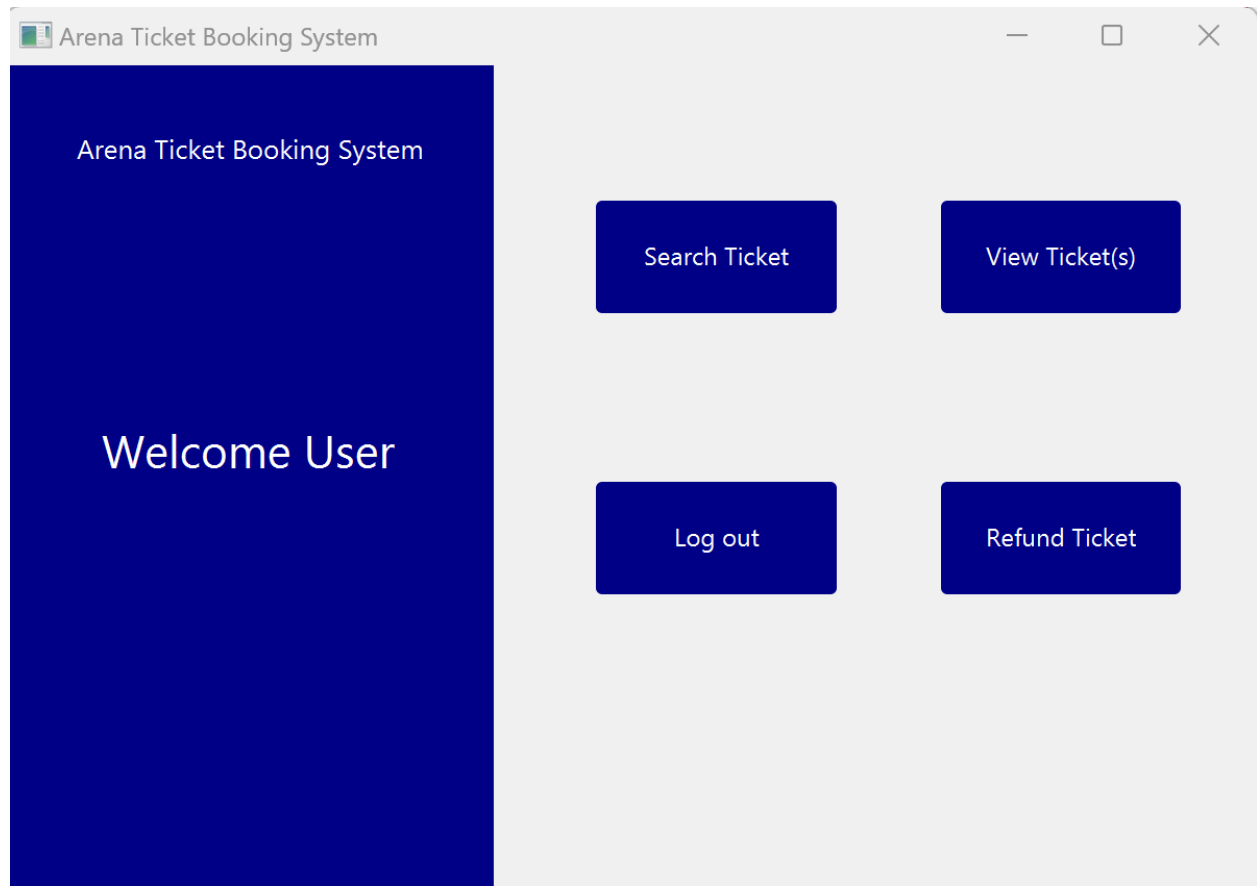Invalid Username

Log In

Arena Ticket Booking System

French

almontel

•••••

Invalid Password

Log In

User Main Menu:

Arena Ticket Booking System

Arena Ticket Booking System

Search Ticket

View Ticket(s)

Welcome User

Log out

Refund Ticket

Technician Main Menu:

Arena Ticket Booking System

Arena Ticket Booking System

Welcome Technician

Create Event

Assign Ticket

Log out

View Available Ticket...
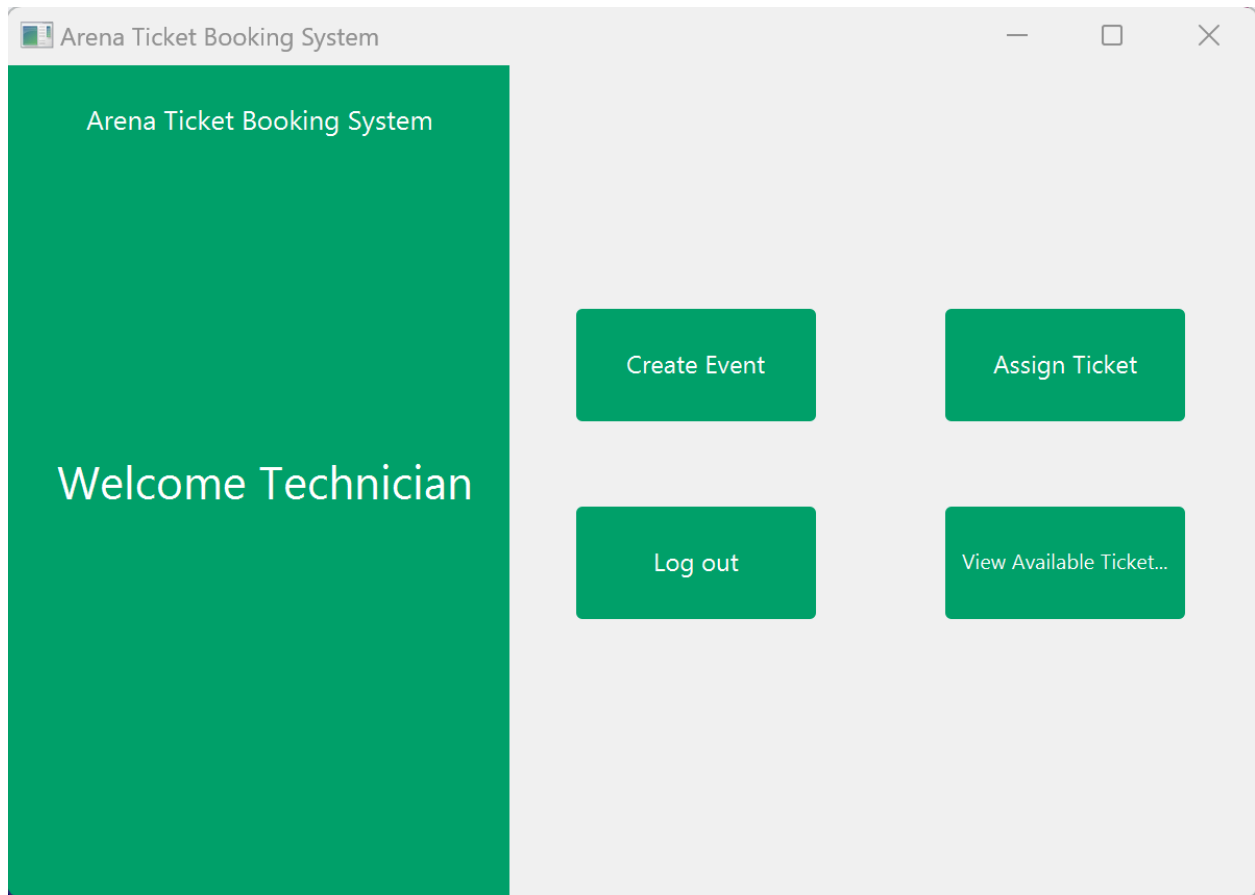
Search Ticket/Book Ticket

Arena Ticket Booking System				— □ ✕

Back   # Available Tickets   Book

Load

TicketID	Event Name	Section	Seat	Price
1	Toronto Raptors VS New York Knicks	1	1	200.0
2	Toronto Raptors VS New York Knicks	1	2	200.0
3	Toronto Raptors VS New York Knicks	1	3	200.0
4	Toronto Raptors VS New York Knicks	1	4	200.0
5	Toronto Raptors VS New York Knicks	1	5	200.0
6	Toronto Raptors VS New York Knicks	2	1	175.0
7	Toronto Raptors VS New York Knicks	2	2	175.0
8	Toronto Raptors VS New York Knicks	2	3	175.0
9	Toronto Raptors VS New York Knicks	2	4	175.0
10	Toronto Raptors VS New York Knicks	2	5	175.0
11	Toronto Raptors VS New York Knicks	3	1	150.0

Available Tickets

Back

Load

Book

TicketID	Event Name	Section	Seat	Price
1	Toronto Raptors VS New York Knicks	1	1	200.0
2	Toronto Raptors VS New York Knicks	1	2	200.0
3	Toronto Raptors VS New York Knicks	1	3	200.0
4	Toronto Raptors VS New York Knicks	1	4	200.0
5	Toronto Raptors VS New York Knicks	1	5	200.0
6	Toronto Raptors VS New York Knicks	2	1	175.0
7	Toronto Raptors VS New York Knicks	2	2	175.0
8	Toronto Raptors VS New York Knicks	2	3	175.0
9	Toronto Raptors VS New York Knicks	2	4	175.0
10	Toronto Raptors VS New York Knicks	2	5	175.0
11	Toronto Raptors VS New York Knicks	3	1	150.0

Arena Ticket Booking System

Available Tickets

[Back] [Load] [Book]

TicketID	Event Name	Section	Seat	Price
1	Toronto Raptors VS New York Knicks	1	1	200.0
2	Toronto Raptors VS New York Knicks	1	2	200.0
3	Toronto Raptors VS New York Knicks	1	3	200.0
5	Toronto Raptors VS New York Knicks	1	5	200.0
6	Toronto Raptors VS New York Knicks	2	1	175.0
7	Toronto Raptors VS New York Knicks	2	2	175.0
8	Toronto Raptors VS New York Knicks	2	3	175.0
9	Toronto Raptors VS New York Knicks	2	4	175.0
10	Toronto Raptors VS New York Knicks	2	5	175.0
11	Toronto Raptors VS New York Knicks	3	1	150.0
12	Toronto Raptors VS New York Knicks	3	2	150.0

Arena Ticket Booking System

Available Tickets

Back
Load
Book

TicketID	Event Name	Section	Seat	Price
1	Toronto Raptors VS New York Knicks	1	1	200.0
2	Toronto Raptors VS New York Knicks	1	2	200.0
3	Toronto Raptors VS New York Knicks	1	3	200.0
6	Toronto Raptors VS New York Knicks	2	1	175.0
7	Toronto Raptors VS New York Knicks	2	2	175.0
8	Toronto Raptors VS New York Knicks	2	3	175.0
9	Toronto Raptors VS New York Knicks	2	4	175.0
10	Toronto Raptors VS New York Knicks	2	5	175.0
11	Toronto Raptors VS New York Knicks	3	1	150.0
12	Toronto Raptors VS New York Knicks	3	2	150.0
13	Toronto Raptors VS New York Knicks	3	3	150.0

The search ticket feature allows users to browse the list of available tickets. When they press load, the tickets will show up. Once they have found the tickets they want to book, they just have to select the tickets and press the Book Button. When the user books a ticket, the ticket is removed from the unassigned tickets and added to the pending/processing tickets which will be processed by the technicians before the user officially owns the ticket.

View Tickets (User)

Arena Ticket Booking System

Back

Load

Your Tickets

TicketID	Event Name	Section	Seat	Price
4	Toronto Raptors VS New York Knicks	1	4	200.0
5	Toronto Raptors VS New York Knicks	1	5	200.0

This allows the user to view the tickets they have booked

Refund Ticket

Arena Ticket Booking System   — ☐ ✕

| Back | | | | Refund |

Your Tickets

| Load | | | | |

TicketID	Event Name	Section	Seat	Price
4	Toronto Raptors VS New York Knicks	1	4	200.0
5	Toronto Raptors VS New York Knicks	1	5	200.0

Arena Ticket Booking System

Back

Refund

Your Tickets

Load

TicketID	Event Name	Section	Seat	Price
5	Toronto Raptors VS New York Knicks	1	5	200.0

Arena Ticket Booking System

Available Tickets

Back

Load

Book

TicketID	Event Name	Section	Seat	Price
16	Toronto Raptors VS New York Knicks	4	1	125.0
17	Toronto Raptors VS New York Knicks	4	2	125.0
18	Toronto Raptors VS New York Knicks	4	3	125.0
19	Toronto Raptors VS New York Knicks	4	4	125.0
20	Toronto Raptors VS New York Knicks	4	5	125.0
21	Toronto Raptors VS New York Knicks	5	1	100.0
22	Toronto Raptors VS New York Knicks	5	2	100.0
23	Toronto Raptors VS New York Knicks	5	3	100.0
24	Toronto Raptors VS New York Knicks	5	4	100.0
25	Toronto Raptors VS New York Knicks	5	5	100.0
4	Toronto Raptors VS New York Knicks	1	4	200.0

This allows users to refund the tickets they no longer want. It displays the user's tickets and allows them to select a ticket and using the refund button, they can refund the ticket. When they refund the ticket, the ticket is moved from the assigned tickets to the unassigned tickets.

Assign Ticket (Process Ticket)

Arena Ticket Booking System

Pending Tickets

[Back]
[Load]
[Assign]

TicketID	Event Name	Section	Seat	Price
4	Toronto Raptors VS New York Knicks	1	4	200.0
5	Toronto Raptors VS New York Knicks	1	5	200.0

Pending Tickets

Back

Load

Assign

TicketID	Event Name	Section	Seat	Price
4	Toronto Raptors VS New York Knicks	1	4	200.0
5	Toronto Raptors VS New York Knicks	1	5	200.0

Arena Ticket Booking System

Pending Tickets

Back

Assign

Load

TicketID	Event Name	Section	Seat	Price
5	Toronto Raptors VS New York Knicks	1	5	200.0

Back

Load

Pending Tickets

Assign

TicketID	Event Name	Section	Seat	Price

No content in table

TicketID	Event Name	Section	Seat	Price
4	Toronto Raptors VS New York Knicks	1	4	200.0
5	Toronto Raptors VS New York Knicks	1	5	200.0

This allows the technicians to process the pending tickets. They are introduced with a list of pending tickets, and they can select which tickets to process. This takes the ticket from the pending tickets list to the assigned tickets list, and it assigns the ticket to the user.

Create Event

Arena Ticket Booking System

Create Event

Back

Event Type :   BasketballGame ▼

Create

Event Name :   Event Name

Please enter event name

This allows the technicians to create events and add the tickets to that event to the available tickets. All they have to do is choose the type of event and the event name and the event will be created and all of the tickets. All the tickets are saved into the system as well as the database. This ensures the data stays in the system even after recompilation.

Depending on the event, a different number of sections and a different number of seats per section.

Basketball Game: 5 Sections of 5 Seats

Hockey Game: 5 Sections of 7 Seats

Concert: 3 Sections of 10 Seats

Spectacle: 5 Sections of 6 Seats

This was implemented thanks to the Abstract Factory Pattern which helps determine what event to create according to the user input.

```java
public class SportsEventFactory extends EventAbstractFactory {    LeanderAlmonte

    @Override  1 usage    LeanderAlmonte
    public Event getEvent(String eventType, String eventName){
        if(eventType == null){
            return null;
        }
        else if(eventType.equalsIgnoreCase( anotherString: "BasketballGame")){
            return new BasketballGame( totalSections: 5, seatsPerSection: 5, eventName);
        }
        else if(eventType.equalsIgnoreCase( anotherString: "HockeyGame")){
            return new HockeyGame( totalSections: 5, seatsPerSection: 7, eventName);
        }
        else{
            return null;
        }
    }
```
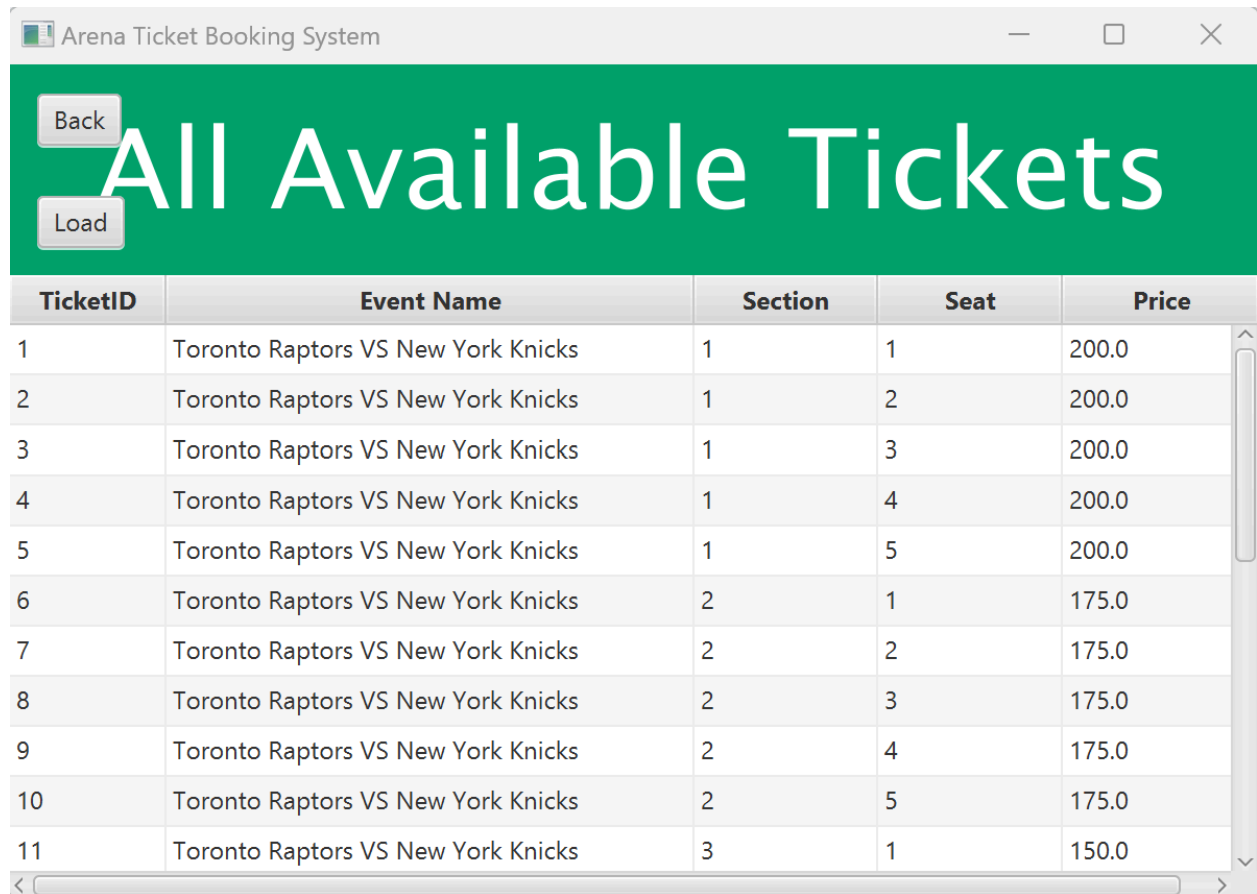
```java
public class EventFactory extends EventAbstractFactory{    LeanderAlmonte +1
//Event Creation Factory Method
    @Override  1 usage    LeanderAlmonte
    public Event getEvent(String eventType, String eventName){
        if( eventType == null){
            return null;
        }
        else if(eventType.equalsIgnoreCase( anotherString: "Spectacle")){
            return new Spectacle( totalSections: 5, seatsPerSection: 6, eventName);
        }
        else if(eventType.equalsIgnoreCase( anotherString: "Concert")){
            return new Concert( totalSections: 3, seatsPerSection: 10, eventName);
        }
        else{
            return null;
        }
    }
```

View All Available Tickets:
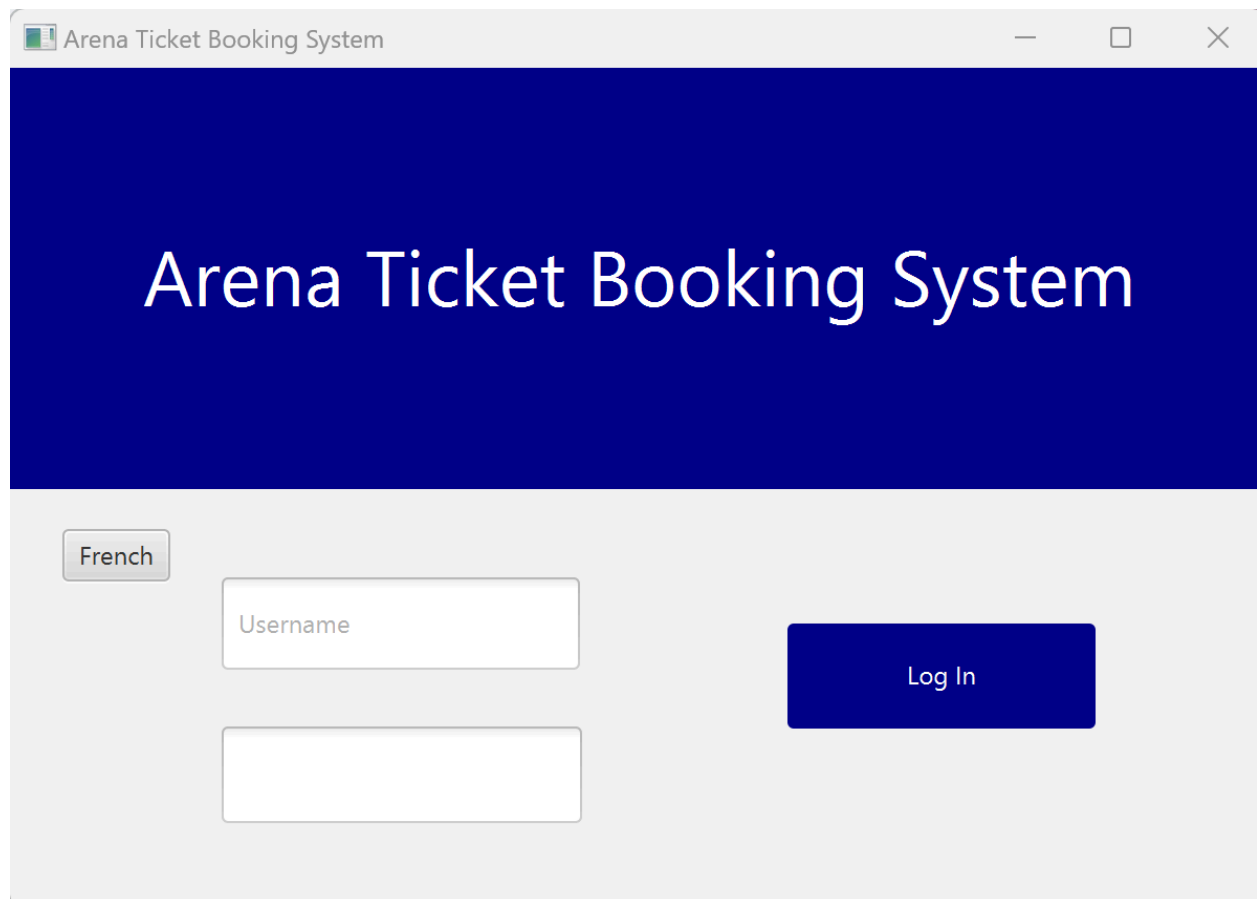
Technicians are able to view all the available tickets in the system.

TicketID	Event Name	Section	Seat	Price
1	Toronto Raptors VS New York Knicks	1	1	200.0
2	Toronto Raptors VS New York Knicks	1	2	200.0
3	Toronto Raptors VS New York Knicks	1	3	200.0
4	Toronto Raptors VS New York Knicks	1	4	200.0
5	Toronto Raptors VS New York Knicks	1	5	200.0
6	Toronto Raptors VS New York Knicks	2	1	175.0
7	Toronto Raptors VS New York Knicks	2	2	175.0
8	Toronto Raptors VS New York Knicks	2	3	175.0
9	Toronto Raptors VS New York Knicks	2	4	175.0
10	Toronto Raptors VS New York Knicks	2	5	175.0
11	Toronto Raptors VS New York Knicks	3	1	150.0

Arena Ticket Booking System

Back

All Available Tickets

Load

Internationalization

Arena Ticket Booking System ___ □ ✕

Arena Ticket Booking System

French

Username

Log In

Système de réservation de billets

Anglais

Nom d'utilisateur

Se connecter

Arena Ticket Booking System

Système de réservation de billets

Anglais

Nom d'utilisateur

Nom et mot de passe invalide

Se connecter

Arena Ticket Booking System

Vos Billets

Retour

Charger

BilletID	Nom de l'événement	Section	Siège	Prix
5	Toronto Raptors VS New York Knicks	1	5	200.0

Arena Ticket Booking System

Système de réservation de billets

Bienvenue Utilisateur

Rechercher Billet

Voir Billet(s)

Déconnection

Rembourser Billet

Système de réservation de billets

Bienvenue Technicien

Créer Événement

Attribuer Billet

Déconnection

Voir Billets Disponibles

Users of the program can use the application in two languages, English and French by using the button on the left. Using two locale objects, and a static class "LocaleManager", the program is able to track throughout the project the language that was selected in the login screen.

Challenges

Version Control:

We were having trouble with maven dependencies and whenever a code would be pushed or pulled, we would get compiler errors, jdk file version errors. This leads us to making new repositories until we figured out how it worked.

We also had some issues with merging but thanks to IntelliJ it made merging commits simple.

GUIs (Displaying Data):

We had some trouble displaying data dynamically throughout the GUIs, for example, whenever the program would execute an action on an arrayList, the displays wouldn't update accordingly.

Complexity of the Project:

We underestimated the complexity of the project, which led us to simplifying our project. We wanted to implement a receipt class,

Unit Testing:

The unit testing was somewhat simple to do however it required the use of multiple functions to allow is to make sure the methods were functioning as designed

Learning Outcomes

Implement Design Patterns and understand their use: The implementation of design patterns helped us further understand their purpose. The design patterns are what really helped build the foundation of this project.

Complexity of using a Database in a project: The implementation of a database in such a project really made us realize how complicated things can get with data manipulation. It made us realize how important but also complicated it is to ensure the data stays consistent throughout the whole project.

Making a project is a long process: We realized how big of a task creating a project from scratch really is. From the brainstorming to the implementation process to the final result is easier said than done. This project really made us realize how much time and effort is required to implement big projects like these.

Programming is not hard, it's just figuring out what are the errors: When we were implementing this project, we were having a lot of issues and errors throughout which lead to a lot of frustration. However, when the solution to the problem is found (through debugging process), it makes programming a lot easier.